BLUETOOTH LOW ENERGY

SEIZURE WRISTBAND CONNECTION

ROBIN YANCEY

SPRING 2016

BLE PROTOCOL STACK

- **Physical Layer:** contains the analog communication circuitry to modulate and demodulate signals and convert the analog signals to digital (figure 1)
  - **Link Layer:** defines roles, eg. who is master, slave, scanner, advertiser
    - **Device Address:** 6 bytes that uniquely identify device among peers (can be registered with IEEE as public address, preprogrammed, or generated on device)
    - **Advertising and Scanning**: when the advertising and scanning channels overlap advertising packet will be received
    - Two types of packets with one data format for advertising and data (31 byte payload)
    - Advertising packets are used to broadcast information and to discover slaves and connect to them
- **Host Controller Interface:** standards for communication across serial interface
- **L2CAP**: manages SM and ATT and takes large packets from upper layers and breaks them into manageable 27 byte maximum transmittable size packets
- GATT and GAP

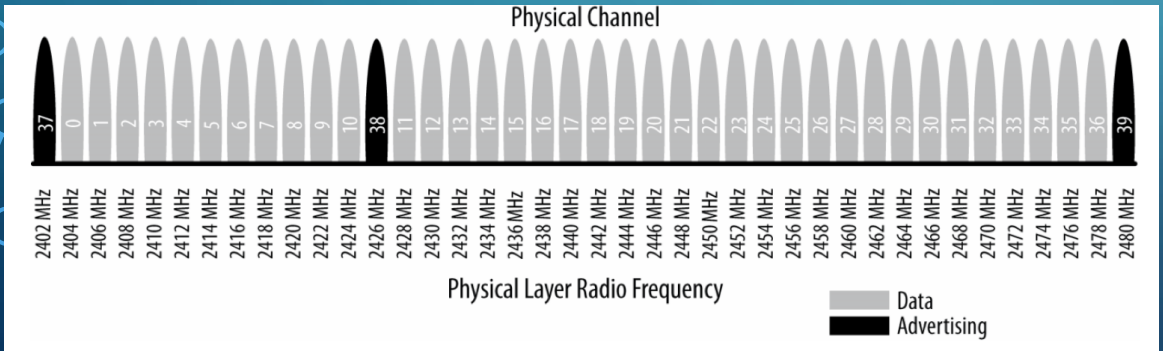

Figure 1. The Physical Layer Radio Frequencies [1]

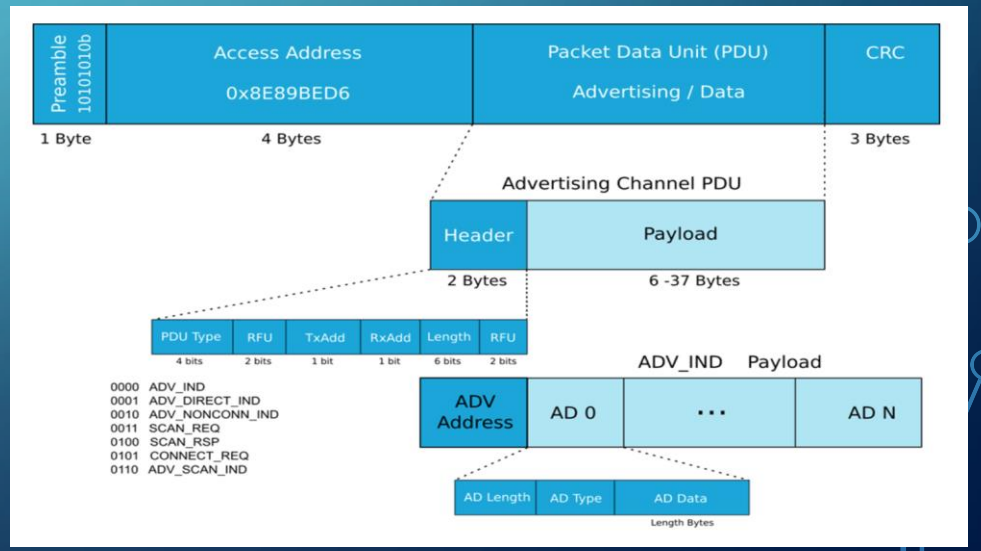

Figure 2. Advertising Payload [2]

GENERIC PROFILES – TOP CONTROL LAYERS OF BLE

- defined by the specification- ensures proper operation between BLE devices from different vendors
- Generic Access Profile (GAP) defines roles, procedures, and modes to allow devices to broadcast data, discover devices, establish connections, manage connections, and security levels, (mandatory for all BLE devices-all must comply) [1] (https://www.bluetooth.com/specifications/assigned-numbers/generic-access-profile)

```
/* Advertisement data */
static uint8_t adv_data[] = {0x1a, 0xff, 0x4c, 0x00, 0x02, 0x15, 0x21, 0x8A,
                             0xF6, 0x52, 0x73, 0xE3, 0x40, 0xB3, 0xB4, 0x1C,
                             0x19, 0x53, 0x24, 0x2C, 0x72, 0xf4, 0x00, 0xbb,
                             0x00, 0x45, 0xc5};

/* scan response data */
static uint8_t scan_rsp_data[] = {0x11, 0x07, 0x1b, 0xc5, 0xd5, 0xa5, 0x02, 0x00,
                                  0x37, 0xaa, 0xe3, 0x11, 0x2a, 0xdc, 0x00, 0xcd,
                                  0x30, 0x57};
```

The scan response data packet (above) allows the user to advertise an additional 27 bytes that follow the advertising data packet

```
/* BLE start advertisement */
if(at_ble_adv_start(AT_BLE_ADV_TYPE_UNDIRECTED, AT_BLE_ADV_GEN_DISCOVERABLE, NULL, AT_BLE_ADV_FP_ANY,
   BEACON_ADV_INTERVAL, BEACON_ADV_TIMEOUT, BEACON_ABSOLUTE_INTERVAL_ADV) != AT_BLE_SUCCESS)
{
```

enum at_ble_adv_type_t

GAP Advertising types.

Enumerator	
AT_BLE_ADV_TYPE_UNDIRECTED	Connectable undirected.
AT_BLE_ADV_TYPE_DIRECTED	Connectable directed.
AT_BLE_ADV_TYPE_SCANNABLE_UNDIRECTED	Scannable undirected.
AT_BLE_ADV_TYPE_NONCONN_UNDIRECTED	Non connectable undirected.
AT_BLE_ADV_TYPE_SCAN_RESPONSE	only used in **AT_BLE_SCAN_INFO** event to signify a scan response
AT_BLE_ADV_TYPE_UNDIRECTED	Connectable undirected.
AT_BLE_ADV_TYPE_DIRECTED	Connectable high duty cycle directed advertising.
AT_BLE_ADV_TYPE_SCANNABLE_UNDIRECTED	Scannable undirected.
AT_BLE_ADV_TYPE_NONCONN_UNDIRECTED	Non connectable undirected.
AT_BLE_ADV_TYPE_DIRECTED_LDC	Connectable low duty cycle directed advertising.
AT_BLE_ADV_TYPE_SCAN_RESPONSE	only used in **AT_BLE_SCAN_INFO** event to signify a scan response

Figure 3. GAP Advertising Type Enumerators in ASF (Atmel Software Framework)

NETWORK TOPOLOGY: BROADCASTING

- The two ways a BLE device can communicate: **broadcasting or connection**
- both method subject to guidelines established by the Generic Access Profile (GAP) [1]
- **Connectionless broadcasting** is sending data out (one-way) to any scanning device (receiver) in listening range picking up the transmitted data.
- An **observer** scans the preset frequencies for non-connectable advertising packets
- **connectionless broadcasting**: user can send to more than one device, only 2 advertising payloads, no security, fast, easy to use
- Figure 4 shows non-connectable advertising packets sent from the SamL21 scanned by the BLE Scanner (iPhone App)

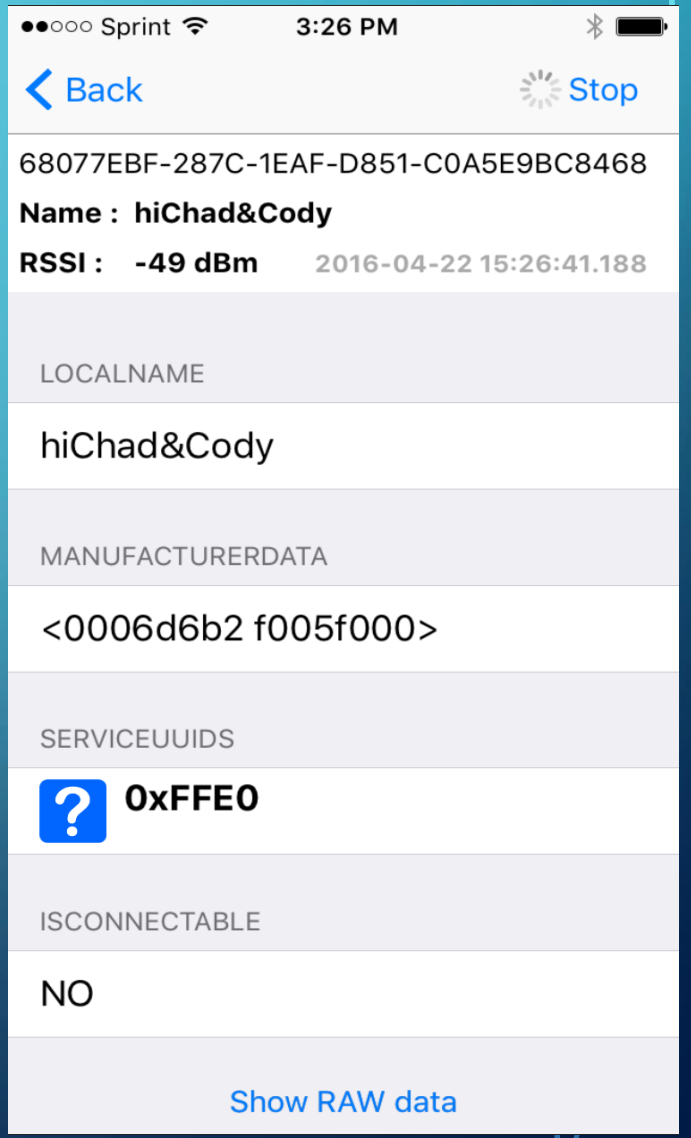

Figure 4. BLE Scanner (iPhone App)

NETWORK TOPOLOGY: CONNECTION

- **connection:** permanent private data exchange between **two** devices
- The **master/central** scans the preset frequencies for connectable packets, initiates and establishes the exclusive connection (when suitable), manages timing and initiates periodical data exchanges (once connected, the peripheral stops advertising) [1]
- **Connection** allows organization of data into characteristics and services through additional Generic Attribute Profile (GATT) protocol layers
- **Services** can have a number of **characteristics** each with their own access rights and descriptive metadata [1]
- This also allows for lower use of power because the peripheral does not need to continually advertise
- (figure 5 128 bit service made available in advertising packet)

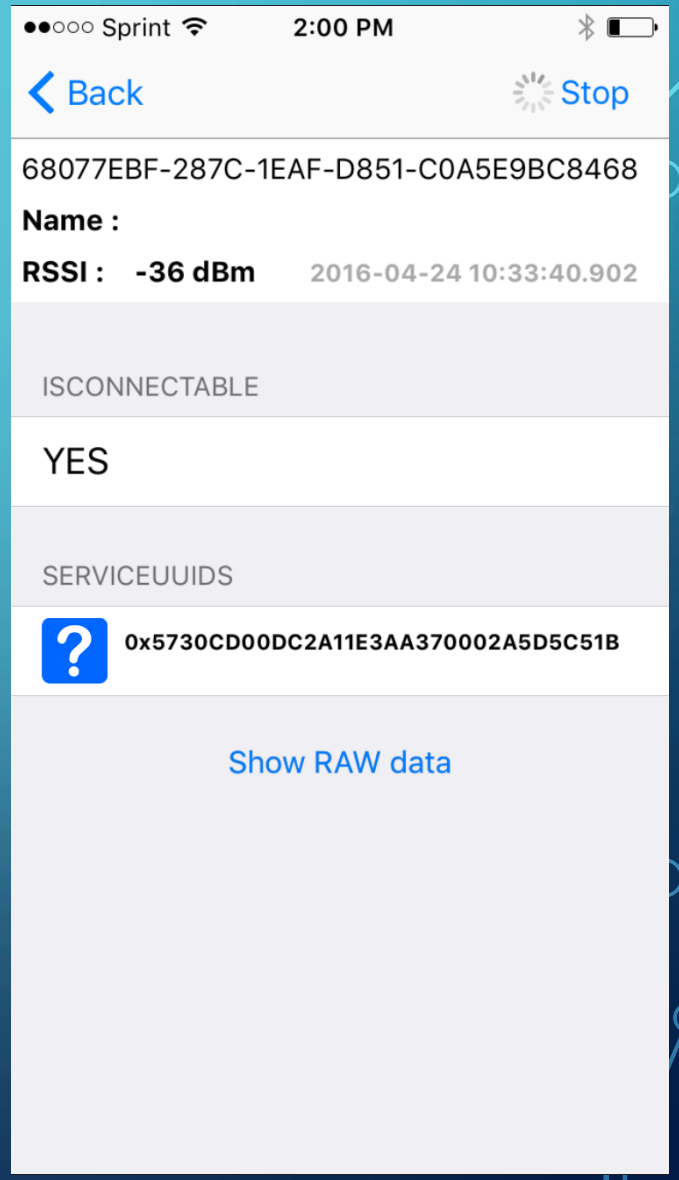

Figure 5. Connectable Advertising Packets

GENERIC PROFILES/TOP LAYERS - (GATT)

- Generic Attribute Profile (GATT) defines the data exchange model and procedures that allow devices to discover, read, write, and push data elements between them [1].
- List of currently adopted services: https://developer.bluetooth.org/gatt/services/Pages/ServicesHome.aspx
- "SIG defined GATT-based profiles provide a predefined set of use-case profiles, based on GATT, that cover all procedures and data formats required to implement a wide range of specific use cases" [1].
- vendors are also allowed to define their own profiles, which can be kept private to the two peers involved in the use case or be published so that other parties can implement them [1].

```
static at_ble_uuid_t service_uuid2 = {AT_BLE_UUID_16 ,
    {0x00, 0x18}};


/* characteristics definitions */
```

```
/* establish peripheral database */
if (at_ble_primary_service_define(&service_uuid2, &service,
NULL, 0, NULL, 0) !=  AT_BLE_SUCCESS)
{
    DBG_LOG("Failed to define the primary service2");
}
```

```
xxxxxxxx-0000-1000-8000-00805F9B34FB
```

Bluetooth Base ID (N/A to vendor-specific UUIDS)

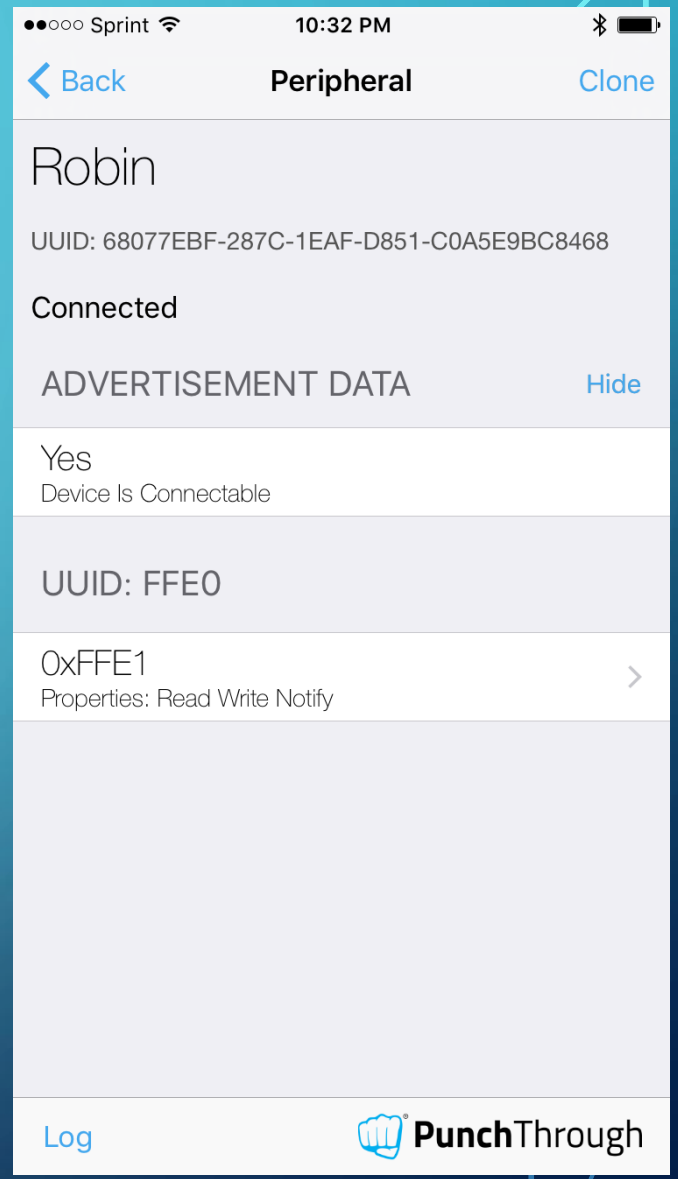

Figure 6. Service and Characteristic UUIDs of the HM 10 and GATT name service 0x0018

GATT PROFILES (CONTINUED)

- servers contains attributes (characteristics), with a 2 byte attribute handle UUID, permissions, and a value. (type = service, handle = characteristic)
- the handle is used to access the value, while the UUID determines the type and nature of data value [1]
- **Characteristics** combine user data with metadata (such as properties, user-visible name, units, ect.)
- Attributes also usually contain data (such as sensor data) stored in RAM
- characteristics contain **descriptors** 8-bit bitfield, (along with the additional two bits in the extended properties descriptor) which contain the operations and procedures that can be used with the characteristic [1].
- **Permissions:** determine whether attribute value can be written to, read, or both (encryption is the security level for accessibility of the client) [1]

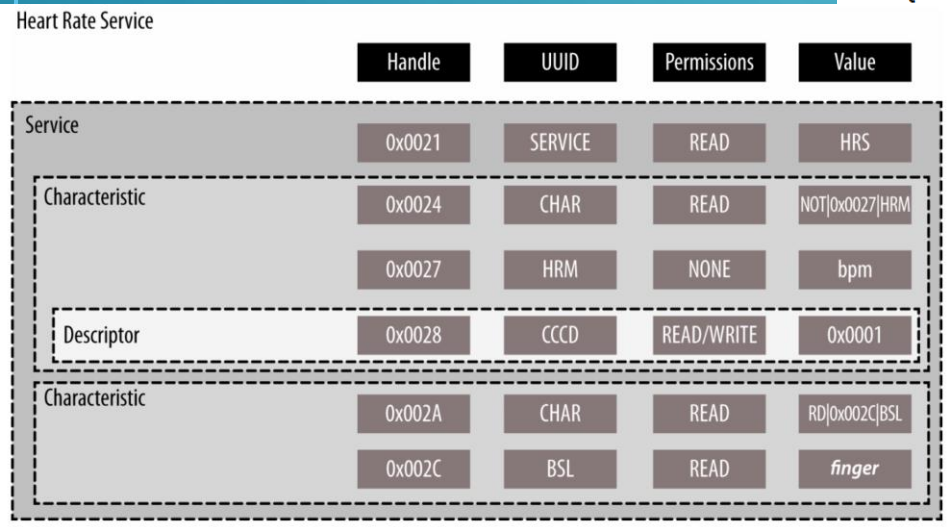

Figure 7. Example of HR Service Characteristic Relationship [1]

```c
/* characteristics definitions */
static at_ble_characteristic_t chars[] = {
    {
0, /* handle stored here */
{   AT_BLE_UUID_16, {0xE1, 0xFF}}, /* UUID */
    AT_BLE_CHAR_READ | AT_BLE_CHAR_WRITE_WITHOUT_RESPONSE | AT_BLE_CHAR_NOTIFY, /* Properties */
    (uint8_t *)"char1", sizeof("char1"), 100, /* value */
    AT_BLE_ATTR_READABLE_NO_AUTHN_NO_AUTHR | AT_BLE_ATTR_WRITABLE_NO_AUTHN_NO_AUTHR, /* permissions */

    NULL, 0, 0, /* user defined name */
    NULL, /* presentation format */
    AT_BLE_ATTR_NO_PERMISSIONS,
    AT_BLE_ATTR_NO_PERMISSIONS,
    AT_BLE_ATTR_NO_PERMISSIONS,
    0,0,0, /* Handles */
```

*See "Advanced Attribute Concepts" and "Service and Characteristic Discovery Features" in text

GATT PROFILES (CONTINUED)

- After adding the services and characteristics of the HM-10 (FFE0 and FFE1), adding the characteristic “properties” to read/write without response/notify” emulates the HM-10 allows the user to connect to Arxterra, as shown in figure 8.
- Changing the “value” in the characteristic definition allows the user to send custom data

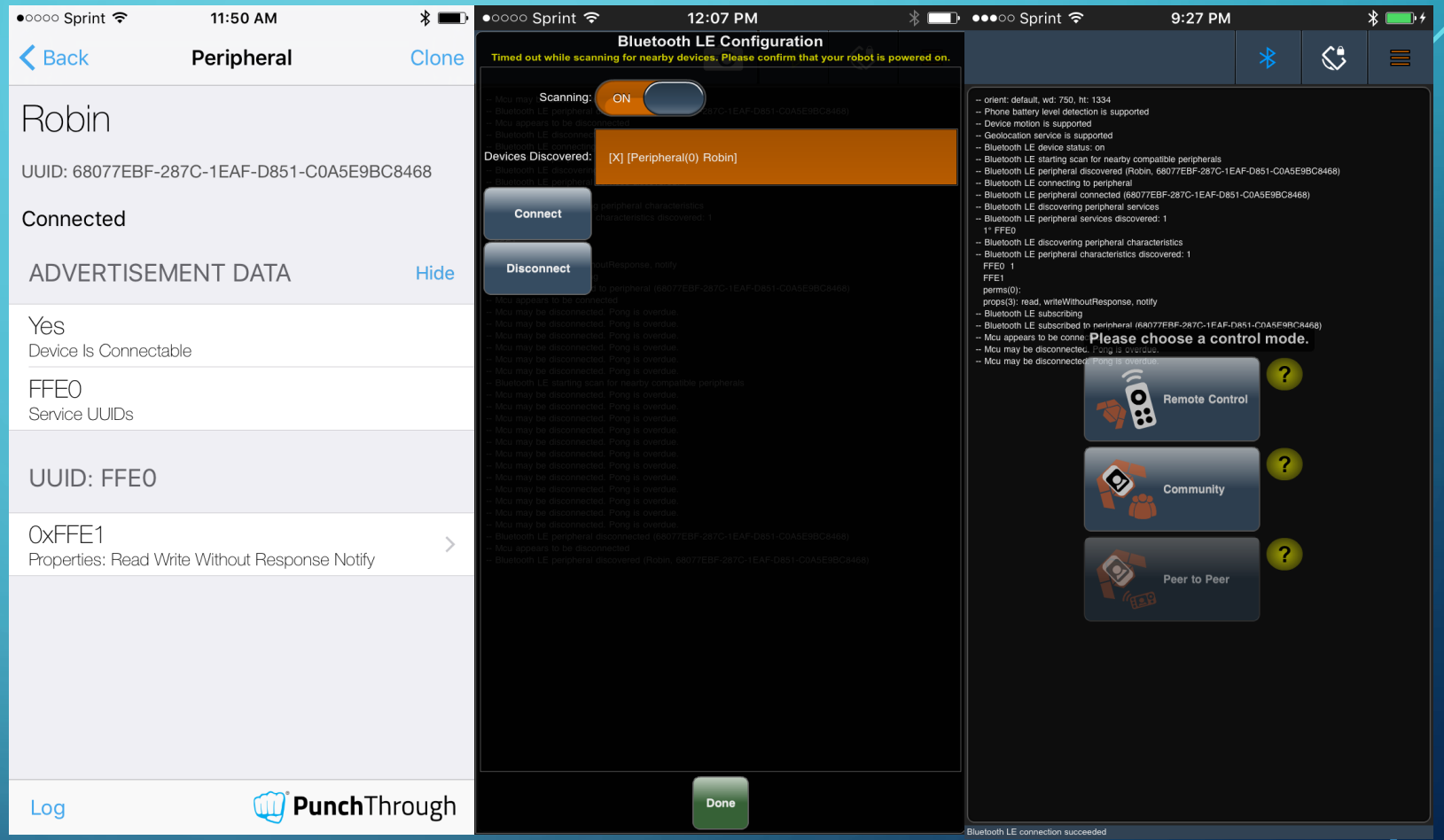

Figure 8. Arxterra Connection

BTLC1000 EXTERNAL FLASH

See "**SAM L21E / SAM L21G / SAM L21J DATASHEET COMPLETE**" [3]

5.3.1. External Flash

ATBTLC1000 Xplained Pro provides a footprint for an external flash (U103), the design is tested with an ISSI *IS25LD020-JNLE* 2Mb flash. By default the flash is connected to the SPI Master/Slave interface of the ATBTLC1000 module, which is also connected to the Xplained Pro extension header.

The SPI Flash master interface of the ATBTLC1000 can also be used to control the external flash by reconfiguring the jumper straps (J109-J112) as below.

External flash Configuration 1: ATBTLC1000 SPI0 peripheral connected (default)
Short straps J109, J110, J111, and J112
Open straps J113, J114, J115, and J116

External flash Configuration 2: ATBTLC1000 SPI flash peripheral connected
Short straps J113, J114, J115, and J116
Open straps J109, J110, J111, and J112

Refer to Design Documentation and the ATBTLC1000-MR110CA datasheet for further reference.

Table 5-5 External Flash Pin Configuration

External flash		Configuration 1, ATBTLC1000 signals			Configuration 2, ATBTLC1000 signals		
Pin	Name	Pin	Name	Function	Pin	Name	Function
1	CE#	12	LP_GPIO_12	SPI0_SSN	21	LP_GPIO_16	SPI Flash SSN
2	SO	14	LP_GPIO_13	SPI0_MISO	23	LP_GPIO_18	SPI Flash RxD
5	SIO	11	LP_GPIO_11	SPI0_MOSI	5	LP_GPIO_3	SPI Flash TxD
6	SCK	10	LP_GPIO_10	SPI0_SCK	4	LP_GPIO_2	SPI Flash SCK

WORKS CITED

1. Townsend, Kevin; Cufí, Carles; Akiba; Davidson, Robert (2014-04-30). Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking (Kindle Locations 675-677). O'Reilly Media. Kindle Edition.
2. *A BLE Advertising Primer*. (2016). Retrieved from:http://www.argenox.com/bluetooth-low-energy-ble-v4-0-development/library/a-ble-advertising-primer/
3. SAML21 Complete. (2016). Retrieved from: http://www.atmel.com/devices/ATSAML21J18B.aspx